Design and Analysis of Algorithms Assignment 4

Harrison Lee, Alex Zhao

February 22, 2019

Question 1. *(6.19)*

Simple observation:
s either ends in x, y, or neither.

Subproblems:
OPT[i] is whether or not s_1 to s_i is an interleaving of x and y, 1 if it is, -1 if it is not.

Recurrence:
Let l_x, l_y, and l_s denote the length of x, y, and s respectively.

$$OPT[i] = \begin{cases} 1 & \text{if } OPT[i - l_x] = 1 \text{ and } s_{i-l_x+1} \text{ to } s_i = x \\ 1 & \text{if } OPT[i - l_y] = 1 \text{ and } s_{i-l_y+1} \text{ to } s_i = y \\ -1 & \text{otherwise} \end{cases}$$

s is an interleaving of x and y if s ends in x or y and the remaining string with that x or y removed is also a interleaving of x and y.

Full Algorithm:
$OPT[i \leq 0] = -1$. Fill in the array from $OPT[1]$ to $OPT[i]$. The final solution is $OPT[l_s]$.
The runtime for this algorithm is $O(l_s)$ (l_s subproblems, each one looking at a constant number of earlier subproblems).

Question 2. *(6.27) Gasoline Re-stocking Scheduling*

Given: Given an initially empty storage tank that can hold L gallons, a price P for each delivery, a cost c for each day a gallon is stored, and g_i gallons sold on day i over $1, ..., n$ days.
Find: Find an algorithm to decide on a schedule for ordering gas shipments.

Algorithm 1. *Find the optimal schedule.*

Proof. Subproblems: $OPT(i)$, is the optimal total expenses ordering shipments up to day i, ending with 0 gas.

Recurrence: $OPT(i) = min\{P + OPT(j) + \sum_{k=i-j}^{i}(k) * (k-1)/2 * c * g_{k+j}$ for all days j from $i - L$ to i

Full Algorithm:

$OPT(0) = 0$ // Base Case

lastRefuelDays = [0] // Track the day of the last refuel for $OPT(index)$

for i = 1, i $\leq$ n, i++ //Go over each day

  currentBestPrice = int.max

  currentBestRefuelDay = -1

  for j = i-L, j $\leq$ i, j++ //Go over days $i - L$ to i. Find $OPT(j)$

    if(j $<$ 0) j = 1 // Skip invalid days

    currentSum = 0

    for k = i-j, k $\leq$ i, k++

      currentSum += $P + OPT(j) + (k) * (k-1)/2 * c * g_{k+j}$

    if currentSum $\leq$ currentBestPrice

      currentBestPrice = currentSum

      currentBestRefuelDay = j

  $OPT(i)$ = currentBestPrice

  lastRefuelDays[i] = currentBestRefuelDay

i = n

refuelingSchedule = []

refuelingQuantities = []

while i != 0 // Collect the final schedule of refueling days

  refuelingSchedule.append(i)

  refuelingQuantities = sum($g_{lastRefuelDays[i]}$ to g_i)

  i = lastRefuelDays[i]

return $OPT(n), refuelingSchedule, refuelingQuantities$

Running Time: $O(n * L * (L * (L-1)/2))$. Each character is compared to the OPT values of the previous L characters, with each of these looking at the next 1 to L days.

□